AMENDMENTS TO THE SPECIFICATION

Please amend the Paragraph beginning on Page 5, line 13, as follows:

The present invention employs cryptographic methodologies in order to secure communications between an administrative console, or host, and remote agents. In this subsection, the basic cryptographic methods employed are described in general terms. Figure 2 illustrates a basic principle underlying cryptographic methodologies. Cryptography is designed to transform plain text information into encoded information that cannot be easily decoded by unauthorized entities. For example, Figure 2 shows a plain text message 202 that includes an English-language sentence. This plain text message can be encrypted by any of various encryption functions *E* ~~1904~~ 204 into a corresponding cipher text message 206 that is not readily interpretable. An authorized user is provided with a decryption function *D* 208 that allows the authorized user to decrypt the cipher text message 206 back to the plain text message ~~1902~~ 202.

Please amend the Paragraph beginning on Page 8, line 16, as follows:

Thus, the techniques of the public key encryption technique can be used to generate digital signatures that can, in turn, be used by a digitally signed message recipient, to verify that a message was sent by the party supplying the digital signature. While it is generally feasible to digitally sign entire, short email messages, it is rather inefficient to digitally sign large amounts of data, such as an executable image. A more efficient way to digitally sign a large amount of data, such as an executable image, is to first digitally hash the large amount of data to a hash value, and then digitally sign the hash value. An efficient hashing function is required that produces a relatively small hash value from a large amount of data in a way that generates large distances in hash-value space between hash values generated from data inputs relatively close together in data-input space. In other words, small changes to input data should widely disperse generated hash values in hash-value space, so that the hash function cannot be deduced systematically. One example of a hash function widely used for this purpose is the

Secure Hash Algorithm ("SHA-1") specified by the Secure Hash Standard~~, available at the web site specified by the URL: http://www.itl.nist.gov/fipspubs/fip180-1.htm~~. The SHA-1 secure hash algorithm generates a 160-bit hash value, called a message digest, from a data file of any length less than $2^{64}$ bits in length. The SHA-1 algorithm is a secure hash because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.

Please amend the Paragraph beginning on Page 10, line 15, as follows:

Next, the agreed-upon secure hash algorithm is used to hash the contents of the firmware-module image 301, excluding the prepended authentication header 308, and the resulting message digest is encrypted with a private encryption key "$PK2_V$" 310 corresponding to the public encryption key "$PK2_P$" 303. The hashing and encryption operations are shown in Figure 3 as step 312. The hashing and encryption of the firmware-module image ~~312~~ 301 produces a digital signature which is appended to the firmware-module image as a verification footer 314.

Please amend the Paragraph beginning on Page 11, line 1, as follows:

Figures 4-7 illustrate the firmware-module-image authentication and verification process used by a calling module to authenticate and verify a firmware module prepared as described with reference to Figure 3. The accessed firmware module is referred to as the ~~"next firmware module."~~ "next firmware module" 408. In Figure 4, the calling firmware module 402 is shown abstractly incorporated within the current execution environment of a secure computer system 404. Note that the calling module includes a stored version of the calling module's public encryption key "$PK1_P$" 406.

Please amend the Paragraph beginning on Page 11, line 15, as follows:

As shown in Figure 6, the calling module 402 next accesses the authentication header, extracting from the authentication header the hashed, encrypted public encryption key "$PK2_P$*" 306 associated with the next firmware module ~~306~~. The

calling module 402 then employs its public key "$PK1_P$" 406 to decrypt the encrypted, hashed public key "$PK2_P$*" to produce a hashed public key "$PK2_P$" 602. Then, the calling module 402 extracts the clear-text public encryption key "$PK2_P$" from the authentication header 308 and hashes the clear-text public encryption key "$PK2_P$" 604, and then compares 606 the extracted hashed $PK2_P$ with the decrypted, hashed $PK2_P$ 602 in order to determine whether or not the next firmware module is an authentic firmware module. If the decrypted, hashed $PK2_P$ 602 is identical to the extracted hashed $PK2_P$ 604, then the next firmware module is an authentic firmware module produced by the firmware module vendor after receiving the hashed, encrypted public encryption key "$PK2_P$*" from the vendor of the calling module 402. After the next firmware module is authenticated, the verification portion of the authentication and verification process ensues, described below with respect to Figure 7. Otherwise, as indicated by the negative arrow 608 in Figure 6, the authentication portion of the authentication and verification process fails, and the calling module 402 may take appropriate action. The calling module 402 may, for example, display an authentication failure message to a console and terminate the bootstrap process.